

`readbackground`

November 29, 2017
17:15

Contents

1 A program to read a background specification from a file	1
--	---

1 A program to read a background specification from a file

\$Id: 506bc1c6d96700f98a720d9a9c11d874f8672e93 \$

This program reads background plasma and neutral source data generated by some other code and translates it into the format needed by DEGAS 2. It is invoked with the command line

`readbackground an_input_file`

This code currently accepts three types of input files:

1. An input file for the original DEGAS code. The file specified on the command line, `an_input_file`, should end with `.d`.
2. A data transfer file generated by the UEDGE code. In this case, `an_input_file` should end in `.u`. This usage is consistent with the original implementation of `readbackground` in which only a single ion species is considered.
3. An input file pointing to a data transfer file generated by the UEDGE code and providing additional descriptive information. This option is selected by having `an_input_file` end in anything other than `.d` or `.u`.

The old DEGAS option has been used infrequently of late, although the DEGAS - DEGAS 2 benchmark remains in the DEGAS 2 examples. Consequently, newer DEGAS 2 features may not be available with this option.

The UEDGE data file is assumed to be in the format corresponding to the UEDGE subroutine `writemcnfile`. M. Rensink has used a BASIS script to create functionally equivalent files that will also work with this program. Presently, this program can only handle runs with two divertor targets, i.e., single null or symmetric double null geometries.

The third option indicated above provides compatibility with multispecies UEDGE runs and allows the user to specify plasma parameters for DEGAS 2 zones filling the volume between the UEDGE mesh and surrounding solid zones (e.g., created with `definegeometry2d`). This input file is of the sort used elsewhere in DEGAS 2. Namely, blank lines, spaces, and lines beginning with the comment character `#` are ignored; comments can also appear at the end of a line.

Each line in the input file is of the form:

keyword arguments

The available keywords are

`uedge_file filename` specifies the name of the UEDGE data file. In this case, `filename` does *not* need to end in `.u`.

ion_species sp1 sp2 sp3 ... provides the list of ion species appearing in the UEDGE run. The order of this list must correspond to the species ordering used by UEDGE in writing the file. The strings provided here should correspond to DEGAS 2 species symbols (i.e., to entries in the *species_sy* array). The code will ignore data corresponding to unrecognized symbols; the code will generate a warning message in those cases. If a species is recognized, but not included in the problem input file for the current DEGAS 2 run (e.g., an impurity species used in UEDGE, but not in DEGAS 2), its density will be used to compute the electron density via quasineutrality. Hence, to guarantee consistency of the electron density in DEGAS 2 and UEDGE, all pertinent species should be added to the DEGAS 2 reference species list. The flow velocity for such species will also be used in computing the total ion flow velocity that is used in the sheath potential expression. If a species in this list is recognized and is in the current problem input file, its density and flow velocity data will be read into the background density and velocity arrays. Its (nonzero) fluxes to the targets will be used to define source groups, one per species per target. These are always assumed to be plate (recycling) sources.

bry_ni sp is n instructs the code to assign ion species *sp* (again, representing a species symbol; it must appear in the *ion_species* keyword and in the current problem input file) a density *n* (in m^{-3}) in zones associated with stratum *i_s*. This stratum number corresponds to those used in *definegeometry2d*'s *stratum* keyword. The stratum number there acts as a label for all zones created out of the polygon specified in conjunction with that keyword. Ensuring uniqueness of the stratum label is ultimately the user's responsibility, although the code may catch some duplications. The stratum labels are contained in the polygon netCDF file generated by *definegeometry2d*; the name of this file must be specified with the *polygon_file* keyword (see below).

bry_vr sp is vr instructs the code to assign ion species *sp* a radial flow velocity *v_r* (in m/s) in zones associated with stratum *i_s*. The cylindrical components of the flow velocity are provided; during DEGAS 2 execution, they resulting vector is rotated to the toroidal angle where the information is needed.

bry_vphi sp is vphi instructs the code to assign ion species *sp* a radial flow velocity *v_φ* (in m/s) in zones associated with stratum *i_s*. The cylindrical components of the flow velocity are provided; during DEGAS 2 execution, they resulting vector is rotated to the toroidal angle where the information is needed.

bry_vz sp is vz instructs the code to assign ion species *sp* a radial flow velocity *v_z* (in m/s) in zones associated with stratum *i_s*. The cylindrical components of the flow velocity are provided; during DEGAS 2 execution, they resulting vector is rotated to the toroidal angle where the information is needed.

bry_ti is Ti instructs the code to assign an ion temperature *T_i* (in eV) to all ion species in the problem (more specifically, all background species except electrons) in zones associated with stratum *i_s*.

bry_te is Te instructs the code to assign an electron temperature *T_e* (in eV) to zones associated with stratum *i_s*.

polygon_file filename specified the name of the polygon netCDF file generated *definegeometry2d*. The information in this file is needed to relate the stratum labels in the *bry_ni*, etc. keywords to the zone numbers.

```
"readbackground.f" 1≡
@m FILE 'readbackground.web'
```

The unnamed module.

```
"readbackground.f" 1.1 ≡  
@f namelist integer  
  
( Functions and Subroutines 1.2 )
```

The main program

```
( Functions and Subroutines 1.2 ) ≡  
program readbackground  
implicit none_f77  
implicit none_f90  
character*FILELEN datafilename  
call command_arg(1, datafilename)  
call process_background(datafilename)  
stop  
end
```

See also sections 1.3 and 1.4.

This code is used in section 1.1.

The main subroutine.

```

⟨ Functions and Subroutines 1.2 ⟩ +≡
subroutine process_background(datafilename)
  implicit none_f77
  zn_common
  pr_common
  bk_common
  so_common
  gi_common
  implicit none_f90
  character*FILELEN datafilename, degasfilename, uedgefilename
  integer uedge, degas, update, geom_modified
@#if 0
  integer i, ii
@#endif

  ⟨ Memory allocation interface 0 ⟩
  call readfilenames
  degas = FALSE
  uedge = FALSE
  /* Find out whether the data file is DEGAS or UEDGE depending on the extension of the file
   name. Have now generalized the UEDGE case to permit some additional options or information
   to be specified in a text file. A DEGAS file is still denoted by a “.d” extension. To use just a
   UEDGE data file by itself, the user would have to give it a “.u” extension. */
  if (index(datafilename, '.d') > 0) then
    degasfilename = datafilename
    degas = TRUE
  @#if 1
  else
    uedgefilename = datafilename
    uedge = TRUE
  @#else
    else if (index(datafilename, '.u') > 0) then
      uedgefilename = datafilename
      uedge = TRUE
    else
      assert('File_name_must_end_in_.d_or_.u' ≡ '_')
  @#endif
  end if

  call read_geometry
  call nc_read_elements
  call nc_read_species
  call nc_read_materials
  call nc_read_reactions
  call nc_read_pmi
  call nc_read_problem
  call setup_back_arrays(geom_modified)
  so_grps = 0
  so_seg_tot = 0
  so_set_run_flags

```

```

so_time_dependent = FALSE

so_gparams_list_size = 0
so_gparams_list_dim = 1
so_params_list_size = 0
so_params_list_dim = 1
so_params_data_size = 0
so_params_data_dim = 1
so_giparams_list_size = 0
so_giparams_list_dim = 1
so_iparams_list_size = 0
so_iparams_list_dim = 1
so_iparams_data_size = 0
so_iparams_data_dim = 1
var_alloc(source_gparameters_list)
var_alloc(source_parameters_list)
var_alloc(source_gparameters_data)
var_alloc(source_parameters_data)
var_alloc(source_giparameters_list)
var_alloc(source_iparameters_list)
var_alloc(source_giparameters_data)
var_alloc(source_iparameters_data)

update = FALSE
if (degas ≡ TRUE) then
    call read_degas_background(degashfilename)
elseif (uedge ≡ TRUE) then
    call read_uedge_background(uedgefilename, update)
endif /* Mindlessly copied this section from defineback following an allocation assertion in
           set_background_sources. Note that we do var_reallocb here so that set_background_sources can
           do an explicit var_realloc once the final so_grps is known. The other (segment-based) arrays,
           however, will still be growing one segment at a time. */
if (update ≡ FALSE) then
    var_reallocb(source_base_ptr)
    var_reallocb(source_num_segments)
    var_reallocb(source_type)
    var_reallocb(source_geometry)
    var_reallocb(source_num_flights)
    var_reallocb(source_num_checkpoints)
    var_reallocb(source_species)
    var_reallocb(source_root_species)
    var_reallocb(source_time_variation)
    var_reallocb(source_num_gparameters)
    var_reallocb(source_num_parameters)
    var_reallocb(source_gparameters_base)
    var_reallocb(source_parameters_base)
    var_reallocb(source_parameters_data_base)
    var_reallocb(source_num_giparameters)
    var_reallocb(source_num_iparameters)
    var_reallocb(source_giparameters_base)
    var_reallocb(source_iparameters_base)
    var_reallocb(source_iparameters_data_base)
    var_reallocb(source_total_current)

```

```
var_reallocb(source_weight_norm)
var_reallocb(source_scale_factor)
end if
call add_cramd_data // So we'll have recomb. rates
call set_background_sources
call init_wt_alias(uedge) // Do likewise in boxgen?
call write_background /* Possibly relabeled vacuum zones. */
if (geom_modified ≡ TRUE)
    call write_geometry
return
end
```

Set up the background data using a DEGAS input file.

```
"readbackground.f" 1.4 ≡
@m increment_params(num_params) so_params.list_size += num_params
if (so_params_list_size > so_params_list_dim) then
    var_realloc(source_parameters_list, so_params_list_dim, so_params_list_size)
    so_params_list_dim = so_params_list_size
end if

⟨ Functions and Subroutines 1.2 ⟩ +≡
subroutine read_degas_background(file)

implicit none_f77
zn_common
pr_common
bk_common
sp_common
el_common
rc_common
so_common
gi_common
sc_common
implicit none_f90

st_decls
⟨ Memory allocation interface 0 ⟩

integer zone, jh, jv, js, jsp, jwall, jseg, seg, iparam      // Local
real xmin, xmax, zmin, zmax, curr_tot
character*FILELEN file
logical new_group

sp_decl(sp_hyd_3)
sp_decl(degas_species_9)
sc_decl(sector)
vc_decl(x1)
vc_decl(x2)

external lookup_sector, neutralize_species
integer lookup_sector, neutralize_species

⟨ pardef.h 0 ⟩
⟨ combal.h 0 ⟩
⟨ comst.h 0 ⟩
⟨ comgeo.h 0 ⟩
⟨ compls.h 0 ⟩
⟨ comflg.h 0 ⟩
⟨ comrfl.h 0 ⟩
⟨ comrat.h 0 ⟩
⟨ comsv.h 0 ⟩
⟨ comstat.h 0 ⟩
⟨ compar.h 0 ⟩

open(unit = diskin, file = file, status = 'old', form = 'formatted')
call inpt(diskin, geometry_symmetry, xmin, xmax, zmin, zmax)
close(unit = diskin)
```

```

sp_hyd_1 = sp_lookup('H')
sp_hyd_2 = sp_lookup('D')
sp_hyd_3 = sp_lookup('T')
nohyds = 0
do js = 1, nphyd
  if (nhydjs ≠ 0)
    nohyds = nohyds + 1
end do
if (nohyds ≡ 1) then
  degas_species1 = sp_hydnhyd1
  degas_species2 = sp_lookup(trim(sp_sy(sp_hydnhyd1)) || '2')
else if (nohyds ≡ 2) then
  degas_species1 = sp_hydnhyd1
  degas_species2 = sp_hydnhyd2
  degas_species3 = sp_lookup(trim(sp_sy(sp_hydnhyd1)) || '2')
  degas_species4 = sp_lookup(trim(sp_sy(sp_hydnhyd1)) || trim(sp_sy(sp_hydnhyd2)))
  if (degas_species4 ≡ 0) then
    degas_species4 = sp_lookup(trim(sp_sy(sp_hydnhyd2)) || trim(sp_sy(sp_hydnhyd1)))
  end if
  degas_species5 = sp_lookup(trim(sp_sy(sp_hydnhyd2)) || '2')
else if (nohyds ≡ 3) then
  degas_species1 = sp_hydnhyd1
  degas_species2 = sp_hydnhyd2
  degas_species3 = sp_hydnhyd3
  degas_species4 = sp_lookup(sp_sy(sp_hydnhyd1) || '2')
  degas_species5 = sp_lookup(trim(sp_sy(sp_hydnhyd1)) || trim(sp_sy(sp_hydnhyd2)))
  if (degas_species5 ≡ 0) then
    degas_species5 = sp_lookup(trim(sp_sy(sp_hydnhyd2)) || trim(sp_sy(sp_hydnhyd1)))
  end if
  degas_species6 = sp_lookup(trim(sp_sy(sp_hydnhyd2)) || '2')
  degas_species7 = sp_lookup(trim(sp_sy(sp_hydnhyd1)) || trim(sp_sy(sp_hydnhyd3)))
  if (degas_species7 ≡ 0) then
    degas_species7 = sp_lookup(trim(sp_sy(sp_hydnhyd3)) || trim(sp_sy(sp_hydnhyd1)))
  end if
  degas_species8 = sp_lookup(trim(sp_sy(sp_hydnhyd2)) || trim(sp_sy(sp_hydnhyd3)))
  if (degas_species8 ≡ 0) then
    degas_species8 = sp_lookup(trim(sp_sy(sp_hydnhyd3)) || trim(sp_sy(sp_hydnhyd2)))
  end if
  degas_species9 = sp_lookup(trim(sp_sy(sp_hydnhyd3)) || '2')
end if
assert(npis ≤ 9) // Assume no impurity species !

assert(sp_sy(pr_background(1)) ≡ 'e')
do zone = 1, zn_num
  if (zn_type(zone) ≡ zn_plasma) then
    jh = zn_index(zone, 1)
    jv = zn_index(zone, 2)
    /* Check that both jh and jv are valid. If not, could set some default */
    assert(jh * jv ≠ 0)
    bk_n(1, zone) = denehvtjh, jv, 1
    bk_temp(1, zone) = electron_charge * tehvtjh, jv, 1
    vc_set(bk_v(1, zone), zero, zero, zero)
  end if
end do

```

```

do js = 1, nohyds
    /* This assumes only singly charged ions. Same assumption made below. */
    jsp = pr_background_lookup(sp_lookup(trim(sp-sy(degas-speciesjs)) || '+'))
    assert(jsp > 1)
    bk_n(jsp, zone) = denihvtjh, jv, 1, js
    bk_temp(jsp, zone) = electron_charge * tihvtjh, jv, 1, js
    vc_set(bk_v(jsp, zone), zero, zero, zero)
end do
end if
end do /* DEGAS only uses cartesian flow velocities */
background_coords = plasma_coords_cartesian

do js = 1, npis
    do jwall = 1, nowals
        new_group =  $\mathcal{T}$ 
        curr_tot = zero
        do jseg = 1, nosegsxzjwall
            curr_tot += abs(currxztjseg, 1, jwall, js)
        end do
        if (curr_tot > zero) then
            do jseg = 1, nosegsxzjwall
                if (new_group) then
                    so_grps++
                    var_realloc(source_base_ptr)
                    var_realloc(source_num_segments)
                    var_realloc(source_type)
                    var_realloc(source_geometry)
                    var_realloc(source_num_flights)
                    var_realloc(source_num_checkpoints)
                    var_realloc(source_species)
                    var_realloc(source_root_species)
                    var_realloc(source_time_variation)

                    var_realloc(source_num_gparameters)
                    var_realloc(source_num_parameters)
                    var_realloc(source_gparameters_base)
                    var_realloc(source_parameters_base)
                    var_realloc(source_parameters_data_base)
                    var_realloc(source_num_giparameters)
                    var_realloc(source_num_iparameters)
                    var_realloc(source_giparameters_base)
                    var_realloc(source_iparameters_base)
                    var_realloc(source_iparameters_data_base)

                    var_realloc(source_total_current)
                    var_realloc(source_weight_norm)
                    var_realloc(source_scale_factor)
                    so_base(so_grps) = so_seg_tot + 1
                    so_nseg(so_grps) = nosegsxzjwall
                    so_nflights(so_grps) = 100
                    so_chkpt(so_grps) = 0
                    so_t_varn(so_grps) = so_delta_fn
                    source_num_gparametersso_grps = 0
                    source_num_parametersso_grps = 0
                end if
            end do
        end if
    end do

```

```

source_gparameters_base_so_grps = so_gparams_list_size
source_parameters_base_so_grps = so_params_list_size
source_parameters_data_base_so_grps = so_params_data_size
source_num_gparameters_so_grps = 0
source_num_iparameters_so_grps = 0
source_giparameters_base_so_grps = so_giparams_list_size
source_iparameters_base_so_grps = so_iparams_list_size
source_iparameters_data_base_so_grps = so_iparams_data_size
/* Assume these are same for all segments: */
if (currxzt_jseg, 1, jwall, js > zero) then
    so_type(so_grps) = so_plate
    so_root_sp(so_grps) = sp_lookup(trim(sp_sy(degas_species_js)) || '+')
    so_species(so_grps) = neutralize_species(so_root_sp(so_grps))
    source_num_parameters_so_grps = 3 // Placeholder only
    increment_params(source_num_parameters_so_grps)
    so_params_list(1, so_grps) = so_param_e_ion_delta
    so_params_list(2, so_grps) = so_param_e_ion_mult
    so_params_list(3, so_grps) = so_param_e_ion_sheath
else
    so_type(so_grps) = so_puff
    so_species(so_grps) = degas_species_js
    so_root_sp(so_grps) = degas_species_js
    source_num_gparameters_so_grps = 2
    so_gparams_list_size += source_num_gparameters_so_grps
    if (so_gparams_list_size > so_gparams_list_dim) then
        var_realloc(source_gparameters_list, so_gparams_list_dim, so_gparams_list_size)
        var_realloc(source_gparameters_data, so_gparams_list_dim, so_gparams_list_size)
        so_gparams_list_dim = so_gparams_list_size
    end if
    so_gparams_list(1, so_grps) = so_gparam_puff_temp
    so_gparams_data(1, so_grps) = t0puff_js * electron_charge
    so_gparams_list(2, so_grps) = so_gparam_puff_exponent
    so_gparams_data(2, so_grps) = one // Hardwired !
end if
assert(so_species(so_grps) > 0)
so_geom(so_grps) = so_surface
so_tot_curr(so_grps) = zero
so_scale(so_grps) = one // Default
seg = so_seg_tot
so_seg_tot += nosegsxz_jwall
var_realloc(source_current, seg, so_seg_tot)
var_realloc(source_segment_ptr, seg, so_seg_tot)
var_realloc(source_segment_rel_wt, seg, so_seg_tot)
var_realloc(source_segment_prob_alias, seg, so_seg_tot)
var_realloc(source_segment_ptr_alias, seg, so_seg_tot)
new_group = F
end if /* Any reason not to exclude segments with zero current? */
if ((xwall(jseg, jwall) ≠ xwall(jseg + 1, jwall) ∨ zwall(jseg, jwall) ≠ zwall(jseg + 1, jwall)) ∧ currxzt_jseg, 1, jwall, js ≠ zero) then
    seg++
    source_current_seg = abs(currxzt_jseg, 1, jwall, js)
    so_tot_curr(so_grps) += source_current_seg

```

```

vc_set(x1, xwall(jseg, jwall), zero, zwall(jseg, jwall))
vc_set(x2, xwall(jseg + 1, jwall), zero, zwall(jseg + 1, jwall))
sector = lookup_sector(x2, x1)
assert(sc_check(sector))
if (so_type(so_grps) ≡ so_plate) then
    assert(source_num_parametersso_grps > 0) // Should be 3 !
    so_params_data_size += source_num_parametersso_grps
    if (so_params_data_size > so_params_data_dim) then
        var_realloc(source_parameters_data, so_params_data_dim, so_params_data_size)
        so_params_data_dim = so_params_data_size
    end if /* Set incident ion description to be 3Te (sheath potential) plus 1 × Eion
           with Eion sampled from a Maxwellian. */
    do iparam = 1, source_num_parametersso_grps
        if (so_params_list(iparam, so_grps) ≡ so_param_e_ion_delta) then
            so_params_data(iparam, seg, so_grps) = const(3.) * bk_temp(1, sector_zonesector)
        else if (so_params_list(iparam, so_grps) ≡ so_param_e_ion_sheath) then
            so_params_data(iparam, seg, so_grps) = zero
        else if (so_params_list(iparam, so_grps) ≡ so_param_e_ion_mult) then
            so_params_data(iparam, seg, so_grps) = one
        end if
    end do
end if
source_segment_ptrseg = sector
else
    so_nseg(so_grps)--
    so_seg_tot--
end if
end do
end do
end do
return
end

```

abs: 1.4.
add_cramd_data: 1.3.
assert: 1.3, 1.4.
background_coords: 1.4.
bdy_ni: 1.
bdy_te: 1.
bdy_ti: 1.
bdy_vphi: 1.
bdy_vr: 1.
bdy_vz: 1.
bk_common: 1.3, 1.4.
bk_n: 1.4.
bk_temp: 1.4.
bk_v: 1.4.
command_arg: 1.2.
const: 1.4.
curr_tot: 1.4.

currxzt: 1.4.
datafilename: 1.2, 1.3.
defineback: 1.3.
definegeometry2d: 1.
degas: 1.3.
degas_species: 1.4.
degasfilename: 1.3.
denehvt: 1.4.
denihvt: 1.4.
diskin: 1.4.

el_common: 1.4.
electron_charge: 1.4.

FALSE: 1.3.
file: 1.4.
FILE: 1.
FILELEN: 1.2, 1.3, 1.4.
form: 1.4.

geom_modified: 1.3.
geometry_symmetry: 1.4.
gi_common: 1.3, 1.4.

i: 1.3.
ii: 1.3.
implicit_none_f77: 1.2, 1.3, 1.4.
implicit_none_f90: 1.2, 1.3, 1.4.
increment_params: 1.4.
index: 1.3.
init_wt_alias: 1.3.
inpt: 1.4.
integer: 1.1.
ion_species: 1.
iparam: 1.4.

jh: 1.4.
js: 1.4.
jseg: 1.4.
jsp: 1.4.
jv: 1.4.
jwall: 1.4.

lookup_sector: 1.4.

namelist: 1.1.
nc_read_elements: 1.3.
nc_read_materials: 1.3.
nc_read_pmi: 1.3.
nc_read_problem: 1.3.
nc_read_reactions: 1.3.
nc_read_species: 1.3.
neutralize_species: 1.4.
new_group: 1.4.
nhyd: 1.4.

nohyds: 1.4.
nosegsxz: 1.4.
nowals: 1.4.
nphyd: 1.4.
npis: 1.4.
num_params: 1.4.
one: 1.4.
plasma_coords_cartesian: 1.4.
polygon_file: 1.
pr_background: 1.4.
pr_background_lookup: 1.4.
pr_common: 1.3, 1.4.
process_background: 1.2, 1.3.
rc_common: 1.4.
read_degas_background: 1.3, 1.4.
read_geometry: 1.3.
read_uedge_background: 1.3.
readbackground: 1, 1.2.
readfilenames: 1.3.
sc_check: 1.4.
sc_common: 1.4.
sc_decl: 1.4.
sector: 1.4.
sector_zone: 1.4.
seg: 1.4.
set_background_sources: 1.3.
setup_back_arrays: 1.3.
so_base: 1.4.
so_chkpt: 1.4.
so_common: 1.3, 1.4.
so_delta_fn: 1.4.
so_geom: 1.4.
so_giparams_list_dim: 1.3.
so_giparams_list_size: 1.3, 1.4.
so_gparam_puff_exponent: 1.4.
so_gparam_puff_temp: 1.4.
so_gparams_data: 1.4.
so_gparams_list: 1.4.
so_gparams_list_dim: 1.3, 1.4.
so_gparams_list_size: 1.3, 1.4.
so_grps: 1.3, 1.4.
so_iparams_data_dim: 1.3.
so_iparams_data_size: 1.3, 1.4.
so_iparams_list_dim: 1.3.
so_iparams_list_size: 1.3, 1.4.
so_nflights: 1.4.
so_nseg: 1.4.
so_param_e_ion_delta: 1.4.
so_param_e_ion_mult: 1.4.
so_param_e_ion_sheath: 1.4.

```
so_params_data: 1.4.  
so_params_data_dim: 1.3, 1.4.  
so_params_data_size: 1.3, 1.4.  
so_params_list: 1.4.  
so_params_list_dim: 1.3, 1.4.  
so_params_list_size: 1.3, 1.4.  
so_plate: 1.4.  
so_puff: 1.4.  
so_root_sp: 1.4.  
so_scale: 1.4.  
so_seg_tot: 1.3, 1.4.  
so_set_run_flags: 1.3.  
so_species: 1.4.  
so_surface: 1.4.  
so_t_varn: 1.4.  
so_time_dependent: 1.3.  
so_tot_curr: 1.4.  
so_type: 1.4.  
source_base_ptr: 1.3, 1.4.  
source_current: 1.4.  
source_geometry: 1.3, 1.4.  
source_giparameters_base: 1.3, 1.4.  
source_giparameters_data: 1.3.  
source_giparameters_list: 1.3.  
source_gparameters_base: 1.3, 1.4.  
source_gparameters_data: 1.3, 1.4.  
source_gparameters_list: 1.3, 1.4.  
source_iparameters_base: 1.3, 1.4.  
source_iparameters_data: 1.3.  
source_iparameters_data_base: 1.3, 1.4.  
source_iparameters_list: 1.3.  
source_num_checkpoints: 1.3, 1.4.  
source_num_flights: 1.3, 1.4.  
source_num_giparameters: 1.3, 1.4.  
source_num_gparameters: 1.3, 1.4.  
source_num_iparameters: 1.3, 1.4.  
source_num_parameters: 1.3, 1.4.  
source_num_segments: 1.3, 1.4.  
source_parameters_base: 1.3, 1.4.  
source_parameters_data: 1.3, 1.4.  
source_parameters_data_base: 1.3, 1.4.  
source_parameters_list: 1.3, 1.4.  
source_root_species: 1.3, 1.4.  
source_scale_factor: 1.3, 1.4.  
source_segment_prob_alias: 1.4.  
source_segment_ptr: 1.4.  
source_segment_ptr_alias: 1.4.  
source_segment_rel_wt: 1.4.  
source_species: 1.3, 1.4.  
source_time_variation: 1.3, 1.4.  
source_total_current: 1.3, 1.4.  
source_type: 1.3, 1.4.
```

source_weight_norm: 1.3, 1.4.
sp_common: 1.4.
sp_decl: 1.4.
sp_hyd: 1.4.
sp_lookup: 1.4.
sp_sy: 1.4.
species_sy: 1.
st_decls: 1.4.
status: 1.4.
stratum: 1.

tehvt: 1.4.
tihvt: 1.4.
trim: 1.4.
TRUE: 1.3.
t0puff: 1.4.

uedge: 1.3.
uedge_file: 1.
uedgefilename: 1.3.
unit: 1.4.
update: 1.3.

var_alloc: 1.3.
var_realloc: 1.3, 1.4.
var_realloca: 1.4.
var_reallocb: 1.3.
var_reallocc: 1.4.
vc_decl: 1.4.
vc_set: 1.4.

write_background: 1.3.
write_geometry: 1.3.
writemcnfile: 1.

xmax: 1.4.
xmin: 1.4.
xwall: 1.4.
x1: 1.4.
x2: 1.4.

zero: 1.4.
zmax: 1.4.
zmin: 1.4.
zn_common: 1.3, 1.4.
zn_index: 1.4.
zn_num: 1.4.
zn_plasma: 1.4.
zn_type: 1.4.
zone: 1.4.
zwall: 1.4.

⟨ Functions and Subroutines 1.2, 1.3, 1.4 ⟩ Used in section 1.1.
⟨ Memory allocation interface 0 ⟩ Used in sections 1.4 and 1.3.
⟨ combal.h 0 ⟩ Used in section 1.4.
⟨ comflg.h 0 ⟩ Used in section 1.4.
⟨ comgeo.h 0 ⟩ Used in section 1.4.
⟨ compar.h 0 ⟩ Used in section 1.4.
⟨ compls.h 0 ⟩ Used in section 1.4.
⟨ comrat.h 0 ⟩ Used in section 1.4.
⟨ comrfl.h 0 ⟩ Used in section 1.4.
⟨ comst.h 0 ⟩ Used in section 1.4.
⟨ comstat.h 0 ⟩ Used in section 1.4.
⟨ comsv.h 0 ⟩ Used in section 1.4.
⟨ pardef.h 0 ⟩ Used in section 1.4.

COMMAND LINE: "fweave -f -i! -W[-ybs15000 -ykw800 -ytw40000 -j -n/
/Users/dstotler/degas2/src/readbackground.web".

WEB FILE: "/Users/dstotler/degas2/src/readbackground.web".

CHANGE FILE: (none).

GLOBAL LANGUAGE: FORTRAN.